Advanced Protected Services

A Concept Paper on Survivable Service-Oriented Systems

Partha Pal, Michael Atighetchi, Joseph Loyall, Andrew Gronosky
Information and Knowledge Technologies Unit
Raytheon BBN Technologies
Cambridge, USA
{ppal,matighet,jloyall,agronosk}@xyz.com

Charles Payne
Cyber Security Group
Adventium Laboratories
Minneapolis, USA
charles.payne@adventiumlabs.org

Robert Hillman
Information Directorate
Air Force Research Laboratory
Rome, USA
robert.hillman@rl.af.mil

***Abstract*— As newer software construction paradigms like service-oriented architecture (SOA) are adopted into systems of critical importance, it becomes imperative that technology and design artifacts exist that can be utilized to raise the resiliency and protection of such systems to a level where they can withstand sustained attacks from well-motivated adversaries. In this paper we describe a sampling of innovative services and mechanisms that are designed for the protection of systems that are based on service-oriented architectures.**

Keywords-survivability, service-oriented architecture

I. INTRODUCTION

The technology landscape of engineering distributed and networked software-based systems has evolved from socket-based network programming to distributed objects and components to web services and service-oriented architectures (SOA). The increased demand on computer-based systems to perform sophisticated tasks and to deploy new functionality quickly, both in civilian, such as the financial and manufacturing sectors, and military domains, is forcing system developers to migrate existing systems and new developments to SOA. At the same time, computer-based systems have become attractive targets of adversaries who aim to benefit by subverting or disrupting the operation of such systems. As a result, there is a need for survivable service-oriented systems—systems that not only tolerate accidental failures, but also continue to deliver an acceptable level of service despite being under attack. In some cases, e.g., critical infrastructure or military systems, systems need to be "ultra-reliable" and it is possible to draw upon additional resources as needed. In other cases, however, resources such as CPU, bandwidth, or the ability to add additional hardware may be limited. In most cases, critical systems have specified timeliness requirements covering various degrees of real time behavior.

Defense against malicious adversaries is an inherently hard problem. An adversary needs to find only one flaw in a system to exploit, whereas the defense needs to identify and address as many as possible. Specific characteristics of SOA approaches add to the challenge, including their dynamism, loose-coupling, and novel messaging and interaction patterns. Moreover, current SOA environments lack the level of sophistication in protection, detection, and adaptation capabilities needed to survive against motivated, well-resourced, and determined adversaries. As a result, current service-oriented systems are at significant risk of corruption, loss of service, and maliciously initiated leakage of information.

To live up to their promised potential, future service-oriented systems will need to *survive* sustained attacks by sophisticated and well-motivated adversaries, something that is not achievable by the way available security solution are currently applied to systems and networks. We argue that *advanced protection*—a synergistic combination of protection, detection, and adaptation capabilities, complemented by the infusion of validated design principles such as defense-in-depth, single point of failure avoidance, containment and isolation—needs to be incorporated into the service-oriented architecture. Furthermore, novel techniques such as automatic generation of configurations and policies from high-level specifications are needed to address the additional risks and vulnerabilities introduced by service-oriented method of system construction.

Toward that end, we have started to develop the necessary technical underpinnings required to make service-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Advanced Protected Services: A Concept Paper on Survivable Service-Oriented Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,Information and Knowledge Technologies Unit,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
1st IEEE International Workshop on Object/component/service-oriented Real-time Networked Ultra-dependable Systems. May 7 2010, Carmona, Spain. U.S. Government or Federal Rights License

14. ABSTRACT
As newer software construction paradigms like service-oriented architecture (SOA) are adopted into systems of critical importance, it becomes imperative that technology and design artifacts exist that can be utilized to raise the resiliency and protection of such systems to a level where they can withstand sustained attacks from well-motivated adversaries. In this paper we describe a sampling of innovative services and mechanisms that are designed for the protection of systems that are based on service-oriented architectures.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**25**	

oriented information systems more robust and resilient against malicious attacks, and to demonstrate the utility of the developed advanced protection techniques in settings that exhibit various threat patterns, resource footprints, and performance requirements (e.g., tactical and tactical edge connected to an enterprise in the military context, or thin and thick clients interacting over various public and private networks in a civilian context). In this paper we describe a selected subset of the key concepts underlying the survivability of service-oriented systems.

The main contribution of the paper is the introduction of advanced survivability concepts into the evolving SOA area. In addition to fostering the dialog between SOA and security researchers over the innovative concepts outlined in this paper, we expect practitioners in the field to benefit by incorporating some of these features into the design and implementation of their systems.

The rest of the paper is organized as follows. In Section II we describe the innovative concepts that we introduce to enable advanced protection in SOA systems. In Section III we describe related work. Section IV concludes the paper.

II. CONCEPTS FOR ADVANCED PROTECTION IN SOA

We argue that in order to achieve the desired level of survivability in the context of a given service-oriented system and its operating environment, one needs a strategic combination of the following:

- *Architecture Enhancements:* Modification of the original organization and interaction pattern of the given system to introduce isolation, containment, redundancy and to enable adaptive behavior.
- *Innovative Defense Mechanis*ms: Insertion of security and adaptive technologies that did not originally exist in the system.
- *Support for Safe and Secure Composition*: Ensuring that systems created by composing architectural elements and supporting defense mechanisms are safe, properly configured and do not introduce residual vulnerabilities.

In this section, we describe representative examples of concepts being developed in each of the above categories that are general enough to have wider applicability in service-oriented settings, irrespective of the specific application or service-oriented platform.

A. Architecture Enhancements

Current SOA-based systems strongly emphasize architectural features that promote reuse through encapsulation and flexibility through composition. The service-oriented community has also adopted a number of security principles, e.g., separation of concerns between policy decision and enforcement [1] through policy decision points (PDPs) and policy enforcement points (PEPs), multi-layer security through WS-Security and traditional demilitarized zones (DMZs), and clustering of application servers. If realized correctly and backed up by reliable implementations of underlying security mechanisms, these principles add to the protection of systems, but each one provides only a point capability and rests on a set of strong environmental assumptions about available resources and attacker sophistication. A survivable service must not make strong assumptions about the environment in which it will operate. Otherwise, an adversary will be able to easily manipulate the environment to violate key assumptions and cause unintended service behavior. For example, a system orchestrated as a workflow of sequential synchronous service invocations might have an implicit assumption about request-response delays. A denial of service attack or a partial failure at some point in the service invocation path can easily violate this timeliness assumption, causing the workflow to complete too late and therefore making results unavailable.

The following paragraphs describe several ways SOAs can be enhanced to significantly increase protection and tolerance over current state of the art.

A single instantiation of a service (and its service container) is a single point of failure (SPOF). Redundancy is traditionally used to mitigate SPOFs and provide load balancing. In addition, advanced redundancy-based protocols such as Byzantine Fault Tolerance [2] have been used to detect and tolerate process corruption. However, simple redundancy is susceptible to common mode failures and easily subverted by malicious adversaries. Dynamic replenishment of redundancy and use of diversity along with redundancy (see Fig. 1a and Fig. 1b) are ways to mitigate that threat.

Dynamic redundancy and diversity are necessary but not sufficient to achieve a high level of survivability. Unlike traditional client-server systems, service consumers across the network do not directly connect to the service instances or replicas but rather to a service broker via an access point or a portal. Although it is common practice for SOA platform components to intercept the service requests for routing and access control purposes in a DMZ, they generally do not inspect and validate request or response payloads, thus leaving the system vulnerable to injection attacks. Furthermore, portals and service brokers do not provide application-level policies for expressing rate and size limits, thereby leaving the system open to denial of service attacks. Furthermore, SOA features like service orchestration introduce dependencies on critical services such as the discovery service, persistence service or the BPEL execution service, which become SPOFs that the designers and developers of services may not realize. Addressing these threats will require the insertion of specific resources and capabilities to absorb the initial blow of a sophisticated attack (see the diverse and redundant

"crumple zone" between the network and the services in Fig. 1c).

Simplistic replenishment of redundancy in the context of create artificial barriers within a system to limit privilege escalation and as shown conceptually in Fig. 1d, limit the spread of attack effects between processes, hosts, and

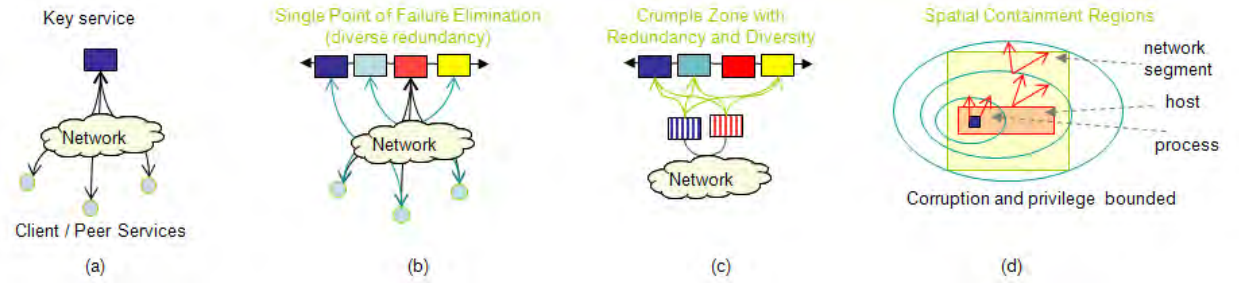


Fig. 1. Organizing Defenses in a Survivability Architecture

service replicas or in the crumple zone is easily overcome by an intelligent and motivated adversary by observing and predicting where the next service replica will be started. To counter that threat, support for adaptive behavior that is unpredictable to the attacker needs to be added to the service-oriented architecture.

Finally, access rights of authenticated users need to be constrained at multiple levels to provide effective defense in depth. Otherwise, an adversary can escalate his privilege and spread attacks throughout the system after gaining an initial entry. SOA provides a level of modularity and plug and play, but in most cases service instantiation and platform implementations are not designed with containment in mind. An entire SOA platform or a key platform component like the JBoss Enterprise Service Bus (ESB) can run as a single Java virtual machine (JVM) process or with similar privileges in a single network partition, making it easy for an adversary to spread attacks

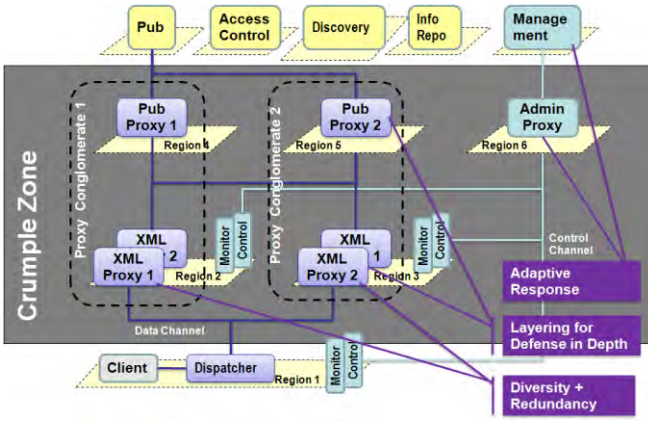


Fig. 2. Design of the Crumple Zone

between components. Similarly data and control channels can be indiscriminately mapped to the same network, leading to situations where attacks on the data channel affect the control channel as well. For these reasons, it is important that a survivable service-oriented system incorporates containment mechanisms by design. Containment regions network segments.

On the basis of the previous discussion, let us turn our focus toward three specific new survivability constructs:

- *Service conglomerates* that eliminate single points of failure,
- *Containment regions* that limit the spread of attacks, and
- *Crumple zones* that strategically combine redundancy with diversity, and interject a defensive layer between two interacting parties.

All three architectural enhancements are based on proven design principles and described in an abstract and vendor neutral manner, and hence generally applicable. Of these three, we describe the crumple zone in more detail because it serves as the integration point of the other concepts.

A **service conglomerate** creates and maintains a set of service instances, some of which may be redundant and possibly diverse variants of one individual service that cooperate closely to defend against a specific threat such as a SPOF or corruption of a high-value service.

A **containment region** creates artificial boundaries that are difficult for an adversary to cross both in terms of time, i.e., expiring unauthorized access after a period of time, and space, i.e., access/compromise in one part or layer of the system will not mean ready access/compromise of other parts and layers of the system.

A **crumple zone** sets up multi-layer intermediaries in the service request-response path that bridges different trust and threat levels, and either stops a class of attacks or provides early warnings of trouble before protected services are affected. The crumple zone uses containment regions and service conglomerates to strengthen the survivability of the outermost interaction surface between clients and services.

Fig. 2 displays a more detailed view of the crumple zone previously shown in Fig. 1c. It shows how the crumple zone is inserted into the interaction pattern between clients at the bottom and services at the top. Proxies contain logic for validating user input, checking attack signatures, and triggering attack effects. To tolerate those effects, the

proxies need to be lightweight and easy to restart. To contain attack effects, proxies are designed to maintain as little state as possible and placed into containment regions, shown as parallelograms.

The crumple zone supports layering of proxies along two axes: horizontal layering, as shown by XML Proxy 1 and XML Proxy 2, provides SPOF elimination through strategic use of redundancy and diversity by having certain proxy functionality, such as XML schema validation, implemented on different implementation substrates, e.g., programming languages and parsers. Vertical layering provides defense in depth by defining proxy chains as a set of functionally different proxy components. For instance, the two proxy service conglomerates shown in Fig. 2 combine XML checking with application-level constraint checking on publication messages sent to a Pub service through the Pub Proxy component.

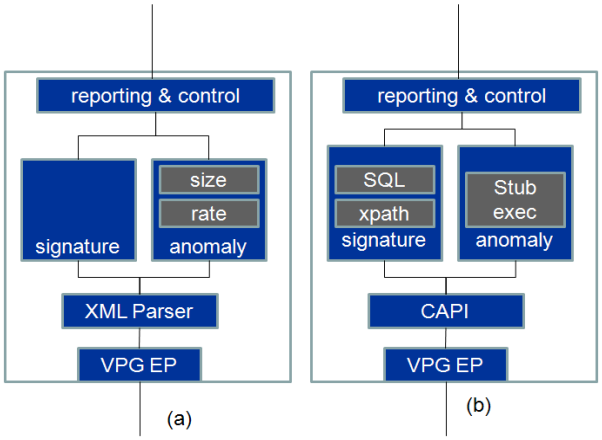


Fig. 3. Two examples of envisioned APS proxy services: (a) interprets and checks XML messages; (b) interprets requests and responses based on knowledge of the application making the request or response

Proper functioning of the crumple zone requires a management and control framework as shown by the backend management service communicating with the crumple zone components to monitor and control proxies through a reverse proxy called the Admin Proxy. The management service contains logic for detecting crashes and corruption of crumple zone components and adaptive response to restart components and initiate dynamic policy reconfigurations, thereby managing the adaptive behavior of the crumple zone.

Various design tradeoffs exist when implementing a crumple zone for a given SOA-based system. We need to relate the amount of redundancy, and diversity, together with the layering depth, to the cost of computing resources and performance overhead. We have started documenting the assumptions, capabilities, and tradeoffs in terms of cost-benefit-risk as part of our methodology for creating survivable designs, which can be used by practitioners to strengthen the survivability of their systems.

We expect to use a combination of logical argument construction, formal methods, and experimentation to validate the claims and benefits of these architectural enhancements. We are investigating the use of domain-specific languages, e.g., Lobster [3], and micro process languages, e.g., Little JIL [4].

B. Novel Defense Mechanisms

The state of the art in SOA security includes mechanisms and services for establishing trusted communication over untrusted networks, e.g., using TLS and WS-Security, policy-based authentication and authorization of services and clients, e.g., using X.509 certificates [5] and XACML[1], and hardened appliances for message checking, e.g., Layer 7 XML gateways [6]. While useful as building blocks, these capabilities are not sufficient to support the architectural survivability concepts described in the previous section. New multi-function defensive mechanisms needed to support the crumple zone concept include proxies that provide protection against Java serialization attacks [7] and can perform anomaly detection capabilities to detect an unexpected increase in rate or size of application-level messages. Furthermore, management of point-to-point SSL [8] connections becomes an issue when dealing with complex interaction patterns spanning many services in a service conglomerate while trying to support end-to-end message integrity across multiple service to service interactions. This section describes two novel defense mechanisms we are investigating in more detail, namely advanced proxy services and service-layer Virtual Private Groups.

Advanced Proxies: Proxy services are becoming more available as part of SOA platforms, for example, AquaLogic [9] includes proxies for services that connect to its ESB. Our envisioned proxies go beyond those of other SOA platforms in that they also contain capabilities to detect and protect against service flood and other injection based attacks. Fig. 3 illustrates the envisioned proxy services. Fig. 3 (a) shows an XML proxy, which interprets the service requests and responses at the level of XML messages and applies signature-based checks as well as size- and rate-based checks. Fig. 3 (b) shows the proxy interpreting the service requests and responses up to the application level and then applying signature and anomaly based checking.

Service Layer Virtual Private Groups (slVPGs): The infusion of architecture enhancements and innovative defense mechanisms described here challenges system self-protection. Service conglomerates and the crumple zone, for example, inject new defenses into the communication path that must integrate seamlessly with existing defenses providing authentication, confidentiality, and integrity guarantees along that path. Not surprisingly, the existing defenses, which include SSL and WS-Security [10], were not designed to facilitate in-transit packet inspection and

directly modifying their use to do so would have many drawbacks, including a higher attack exposure for the new defenses and more complex key management overall. For this reason, we explore the use of slVPGs to guarantee authentication, confidentiality, and integrity between the client and the protected service. The slVPG resembles existing defenses in operation but shares its cryptographic credentials transparently with authorized intermediaries like the crumple zone proxies. The slVPG allows those proxies to peer into protected data streams without affecting the end-to-end guarantees on those streams. The slVPG can also make the use of service conglomerates transparent to the client since all service instances share the same slVPG credentials. The slVPG relies on a protected and robust key distribution scheme, which we will integrate into our larger defense management strategy. System and component tests will include efforts to observe and circumvent slVPG key sharing.

We previously invoked VPGs at the network layer [11] and we observed many benefits from their use (see [12]). However, unlike that earlier incarnation, which relied on a closed, proprietary implementation and protocol [13], our implementation of slVPGs will follow the practice described in [14], where we leveraged the features of robust, open source standards and technologies. The slVPG will provide similar guarantees to existing defenses while facilitating the use of robust defenses for survivability.

C. *Support for Composition-Oriented Software Construction*

Service-orientation encourages decoupled development of functionality as services which then need to be composed together to perform organization-level tasks. For example, an order fulfillment system might be composed of services for order entry and payment validation. This composition-oriented method of software construction needs to integrate and interoperate with the advanced survivability features described earlier, many of which are implemented as services, and some (mostly the architectural enhancements) impose specific constraints on how services are allowed to interact with each other.

The challenge is twofold: first, to integrate services, including defense services, in a way that does not inadvertently open new vulnerabilities; and second, to deploy services in way that the combined resource footprint and run-time cost of business services and survivability services falls well within system requirements. Examples of how improper composition may undermine the protection and continued functioning of the system include:

- A service that was behind a firewall becomes connected to other hosts on the unprotected network, creating a bypass around the firewall that attackers can exploit.
- Authentication is only present on particular entry points into the system but not on others, opening up backdoors.
- Incorporating authentication without encryption of data causes sensitive application data to be sent "in the clear". A special case of this is composition of a distributed password authentication service without encryption, where passwords are sent or stored in plain text.
- Byzantine fault tolerance requires several replicas and complex agreement protocols, which might push the request-response time to an unacceptable level.
- Use of diversity to avoid common mode failures requires multiple operating systems and platforms. The organization now needs system administration expertise for all of these platforms and the increase in complexity might lead to misconfiguration of parts of the system.

SOA developers and system integrators currently do not have much support for validating and checking their deployment configurations for errors and unsafe patterns. To address this issue, we are exploring the following innovative concepts:

- *Safe and Unsafe Composition Patterns*: Define tell-tale characteristics of inherently unsafe compositions.
- *Characterization of Defense Mechanisms and Services Using Metadata:* Empirically determine metadata capturing resource footprint and performance overhead and associate them with the defense mechanisms and services in a secure way.
- *Analysis and Automated Configuration Generation:* Analyze the configuration of a collection of configurable elements in a SOA system to look for consistency or unsafe patterns and generate consistent configuration from high level specifications.

In the following paragraphs we briefly summarize the basic ideas and resulting benefits for each of these concepts.

In the **safe and unsafe composition patterns** thrust, the idea is to construct and study a number of composition use cases and identify patterns that are inherently unsafe and should be avoided. With a catalog of "unsafe" composition patterns, SOA designers and integrators should at least be able to identify and eliminate them from their designs and systems, thereby achieving a basic level of protection.

By **characterizing of defense mechanisms and services using metadata**, we are making the "contract" between the defense provider, i.e., the defense mechanisms and services, and the consumer, i.e., the system which integrates the defense, more explicitly aware of the resource requirement and performance overhead. With such metadata attached to the defense mechanisms and security services, it is

conceivable that a pre-deployment time tool will analyze the intended service descriptor and configuration settings of the relevant elements in the target SOA container, and present an estimate of whether any resource or performance requirements are at risk.

In **analysis and automated configuration generation**, we are leveraging prior work [12] [14] to generate a consistent set of configurations based on a high-level specification such as conversations. A conversation describes authorizations for a set of consumers to engage in specific service interactions. In this context we use the term "authorizing" in a general sense to include authentication as the basis to authorize entities, and additional protection requirements such as confidentiality on the authorized interactions. Each conversation clarifies the desired permissible relationships between the named consumers and providers with regard to named services. A survivable SOA system will need to integrate multiple defense mechanisms and security services, each of which will have their own set of configurable parameters. It has been argued that incorrect configurations account for a large proportion of security vulnerabilities today. Automated configuration generation and checking for predefined safety and consistency properties is a decisive step in addressing this problem.

III. Related Work

Research in survivability and intrusion tolerance, ongoing work in standardizing SOA security, and best-of-breed security available in the SOA market place are all relevant for the work described in this paper. In the commercial space of SOA security products, related work includes hardened security appliances, like the Layer7 XML appliance [6] and F5 BIG-IP ASM [15] as well as endpoint security monitoring systems, e.g., Cisco CSA [16]. Commercial products started supporting a number of OASIS security standards, in particular WS-Security [10] and related subparts such as SAML [17] , XML Signature [18], and XML Encryption [19].

Many researchers exploring adaptive cyber defense, including the authors, have also developed special purpose architectures and supporting mechanisms for intrusion detection and response, intrusion tolerance, and graceful degradation. Many of the design principles and defense mechanisms underlying the advanced survivability concepts described in this paper have been validated through multiple rounds of R&D and evaluation. We summarize a few of these precursor efforts.

The ITUA [20] project developed technology and system design techniques for building information systems that will tolerate, i.e., continue to function without violating program and data integrity, a specific class of attacks, namely, the attacks that introduce corruption in communication and application level interaction in distributed object applications. In addition to corruption tolerant algorithms, ITUA developed architecture for managing distributed object replicas and the hosts on which they run.

The Willow architecture [21] achieves intrusion tolerance using a combination of disabling of vulnerable network elements when a threat is detected or predicted, replacing failed system elements, and reconfiguring the system if non-maskable damage occurs. Willow uses its own event-notification service as the control mechanism of its scalable architecture.

Dependable Intrusion Tolerance (DIT) [22] comprises functionally redundant HTTP commercial off the shelf servers. These servers run on diverse operating systems and platforms, use hardened intrusion-tolerant proxies that mediate client re-quests and verify the behavior of server and other proxies, and include monitoring and alert-management components based on the EMERALD Intrusion Detection System. The system adapts its configuration dynamically in response to intrusions and other faults.

Malicious and Accidental Fault Tolerance for Internet Applications (MAFTIA) [23] is a European project developing an open architecture for transactional operations on the Internet. MAFTIA models a successful attack on a security domain, leading to corruption of processes in that domain, as a fault; the architecture then exploits approaches to fault tolerance that apply regardless of whether the faults are due to accidents or malicious acts. MAFTIA is explicitly middleware-based and provides both protection from and tolerance of intrusions

The Saber [24] system uses several mechanisms including intrusion detection, automatic code patching, process migration, and filtering of distributed denial-of-service floods for defense, but focuses primarily on server avail-ability.

As part of the DARPA OASIS Dem/Val program and BBN's DPASA project [11], we designed and evaluated a high watermark survivability architecture that provides strong guarantees for attack tolerance and survival, intrusion detection, and situational awareness.

The goal of the CSISM project [25] was to develop automated reasoning mechanisms that when incorporated in the survivability architecture will minimize the role of human experts and pave the way for truly self-regenerative survivable systems. CSISM implemented multi-layer reasoning, with fast reaction rules designed to take effective defensive actions within 250 ms of attack initiation, and a more deliberate cognitive reasoning process based on interpretation and hypothesis generation, response selection, and learning to take system wide defense actions.

IV. Conclusion

Securing critical service-oriented systems is an elusive goal as it requires eliminating every vulnerability, known

and unknown, past, present, and future. Determined adversaries only need to find a single vulnerability to successfully exploit, meaning that the odds are stacked in their favor. We suggest a focus on building systems that are survivable instead, using technologies that combine protection, detection, and adaptation capabilities. The wide range of application and network contexts in which service-oriented systems are expected to be deployed implies that there is no one single architecture or configuration to achieve survivability. Designing a survivable system is necessarily a careful balance between requirements, capital and operating expense of the added security measures and the perceived benefits determined based on imperfect and often subjective evaluation of security, and its effect on other requirements such as performance and system size.

The advanced protection concepts we described in this paper work with, utilize, and augment existing security standards and protection mechanisms, and are designed to be tailored for specific contexts. In addition to providing key architectural and protection concepts, it is important to characterize the cost and benefit of the architectural constructs, security services and defense mechanisms so that designers can make informed choices about including them in their systems.

The major contribution of the work described here is the formulation of advanced protection concepts and a methodology to apply them to service-oriented systems. We expect that the concepts, the methodology, and instantiation and validated use cases of a representative sample of the technologies described in this paper in the form of a survivable service-oriented system will provide a foundational basis for engineering dependable service-oriented systems.

ACKNOWLEDGMENT

This material is based upon work supported by the Air Force Research Laboratory under Contract No. FA8750-09-C-0216. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL. The authors wish to thank the other APS team members including Jon Webb and Matt Gillen at BBN, and Dick O'Brien and John Ghode at Adventium.

REFERENCES

[1] Tim Moses. (1 Feb 2005) eXtensible Access Control Markup Language (XACML) Version 2.0.

[2] R Shostak, M Pease, and L Lamport, "The Byzantine Generals Problems," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382-401, July 1982.

[3] P White. (2008) SELinux Developers Summit - Security Configuration Domain Specific Language. [Online]. http://selinuxproject.org/files/2008_selinux_developers_summit/2008_summit_white.pdf

[4] Leon Osterweil et al., "Experience in Using a Process Language to Define Scientific Workflow and Generate Dataset Provenance," in *ACM SIGSOFT 16th International Symposium on Foundations of Software Engineering*, 2008, pp. 319-329.

[5] (2005) ITU-T Recommendation X.509 Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 08/05. [Online]. http://www.itu.int/rec/T-REC-X.509-200508-I

[6] Layer7. (2010, January) XML Firewall. [Online]. http://www.layer7tech.com/main/products/xml-firewall.html

[7] Mark Schoenefeld, "Presentation on J2EE Penetration Testing," , 2006.

[8] T Dierks and E Rescorla. (2008, August) RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2. http://tools.ietf.org/html/rfc5246.

[9] BEA. (2010, Jan) BEA AquaLogic. [Online]. http://www.oracle.com/bea/index.html?CNT=index.htm&FP=/content/products/aqualogic/

[10] OASIS. (2006, February) WS-Security 2004. [Online]. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[11] J. Chong, P. Pal, M. Atighetchi, P. Rubel, and F. Webber, "Survivability Architecture of a Mission Critical System: The DPASA Example," in *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005, pp. 495-504.

[12] Paul Rubel, Michael Ihde, Steven Harp, and Charles Payne, "Generating Policies for Defense in Depth," in *21st Annual Computer Security Applications Conference*, Tucson, Arizona, December 5-9 2005.

[13] M Carney, R Hanzlik, and T Markham, "Virtual Private Groups," in *3rd Annual IEEE Information Assurance Workshop*, 2002.

[14] Richard O'Brien and Charles Payne, "Virtual Private Groups for Protecting Critical Infrastructure Networks," in *Cybersecurity Applications and Technology Conference for Homeland Security*, Washington DC, 2009, pp. 118-123.

[15] F5. (2010, January) BIG-IP ASM. [Online]. http://www.f5.com/products/big-ip/product-modules/application-security-manager.html

[16] Cisco. (2010, January) Cisco Security Agent. [Online]. http://www.cisco.com/en/US/products/sw/secursw/ps5057/index.html

[17] N. Ragouzis et al. (2008, March) Security Assertion Markup Language (SAML) V2.0 Technical Overview. [Online]. http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[18] W3C. (2008, June) XML Signature. [Online]. http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/

[19] W3C. (2002, December) XML Encryption. [Online]. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/

[20] P. Pal et al., "An architecture for adaptive intrusion-tolerant applications," *Software: Practice and Experience*, vol. Volume 36, no. Issue 11-12 (September - October 2006), pp. 1331-1354, 2006.

[21] Dennis Heimbigner, Alexander Wolf, Antonio Carzaniga, Jonathan Hill, Premkumar Devanbu, and Michael Gertz John Knight, "The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications," in *Proc. Int'l Conf. Dependable Systems and Networks (DSN 02)*, 2002, p. C.7.1–C.7.8.

[22] Magnus Almgren, Steven Cheung, Yves Deswarte, Bruno Dutertre, Joshua Levy, Hassen Saidi, Victoria Stavridou, and Tomas E. Uribe. Alfonso Valdes, "An Architecture for an Adaptive Intrusion Tolerant Server," in *Proc. Security Protocols Workshop, LNCS*, 2002.

[23] N. F. Neves, and M. Correia. P. Verissimo, "The Middleware Architecture of MAFTIA: A Blueprint," in *Proc. 3rd IEEE Info. Survivability Workshop*, 2000.

[24] Janak Parekh, Philip N. Gross, Gail Kaiser, Vishal Misra, Jason Nieh, Dan Rubenstein, and Sal Stolfo. Angelos D. Keromytis, "A Holistic Approach to Service Survivability," in *Proc. ACM Workshop on Survivable and Self-Regenerative Systems, ACM Press*, 2003, pp. 11-20.

[25] Partha Pal, Franklin Webber, Paul Rubel, and Michael Atighetchi D. Paul Benjamin, "Using A Cognitive Architecture to Automate Cyberdefense Reasoning," in *Proceedings of the 2008 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS 2008)*, 2008.

Advanced Protected Services

A Concept Paper on Survivable Service Oriented Systems

Presented by Aaron Adler

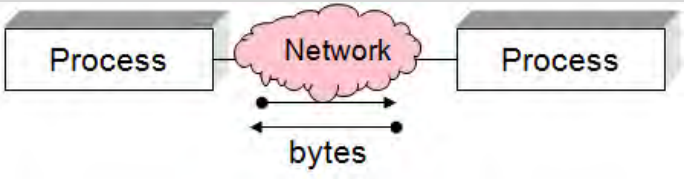
Partha Pal, Michael Atighetchi, Joseph Loyall, Andrew Gronosky,
Charles Payne, Robert Hillman

This work is supported by the US Air Force Research Laboratory

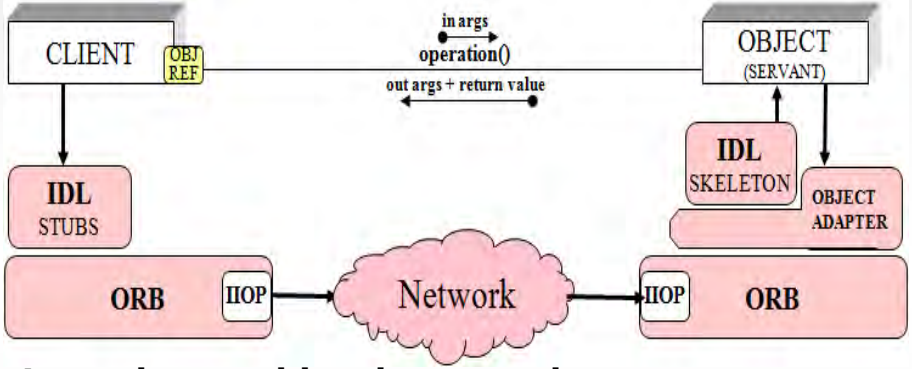
Outline

- Distributed System Technology Landscape
- Survivable Systems
- APS Concepts
- Achievements So Far
- Next Steps and Conclusion

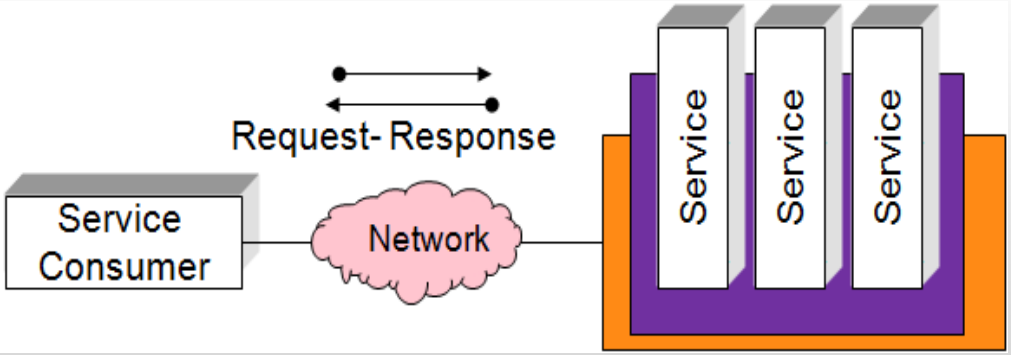
Evolving Technology Landscape



Sockets– transporting bytes

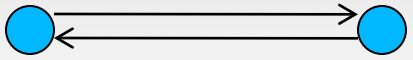


Location and implementation transparent distributed objects

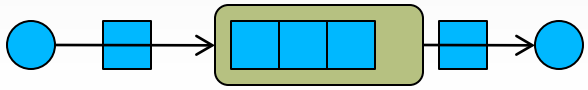


SOA Loosely coupled interaction with container deployed services

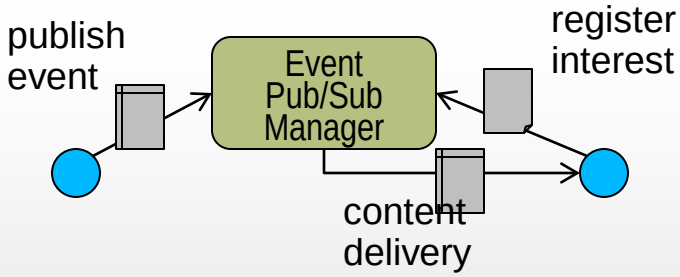
- More functionality pushed down to the "platform" making it more complex to configure and manage
- Various interaction paradigms



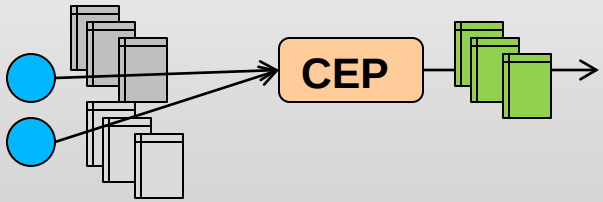
Request-Response



Message/Queues

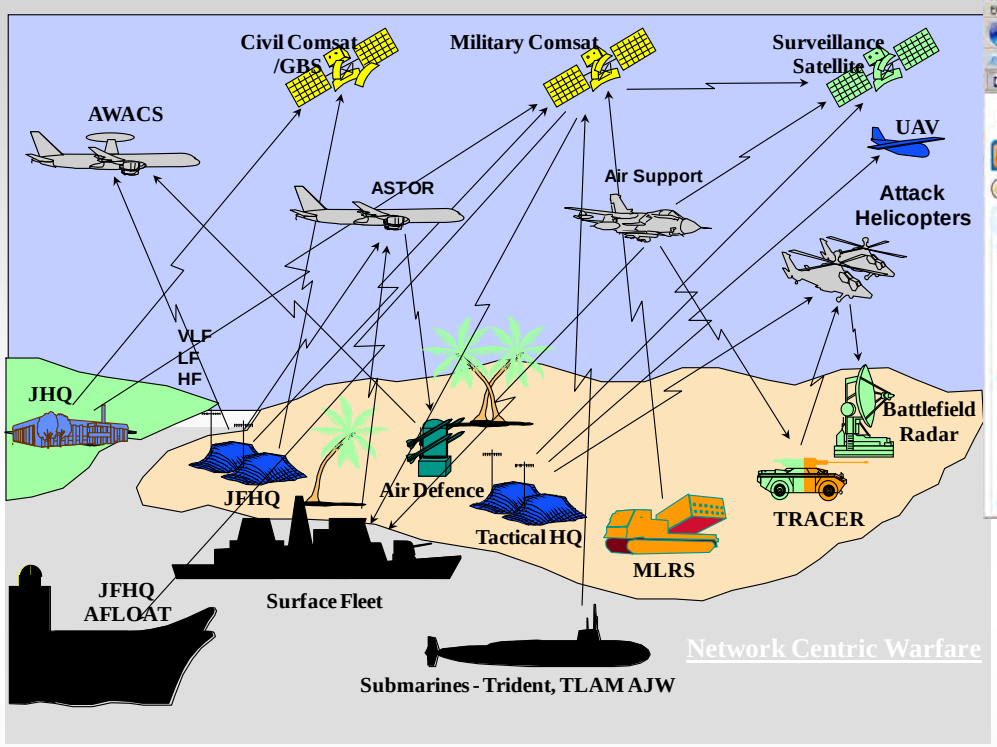


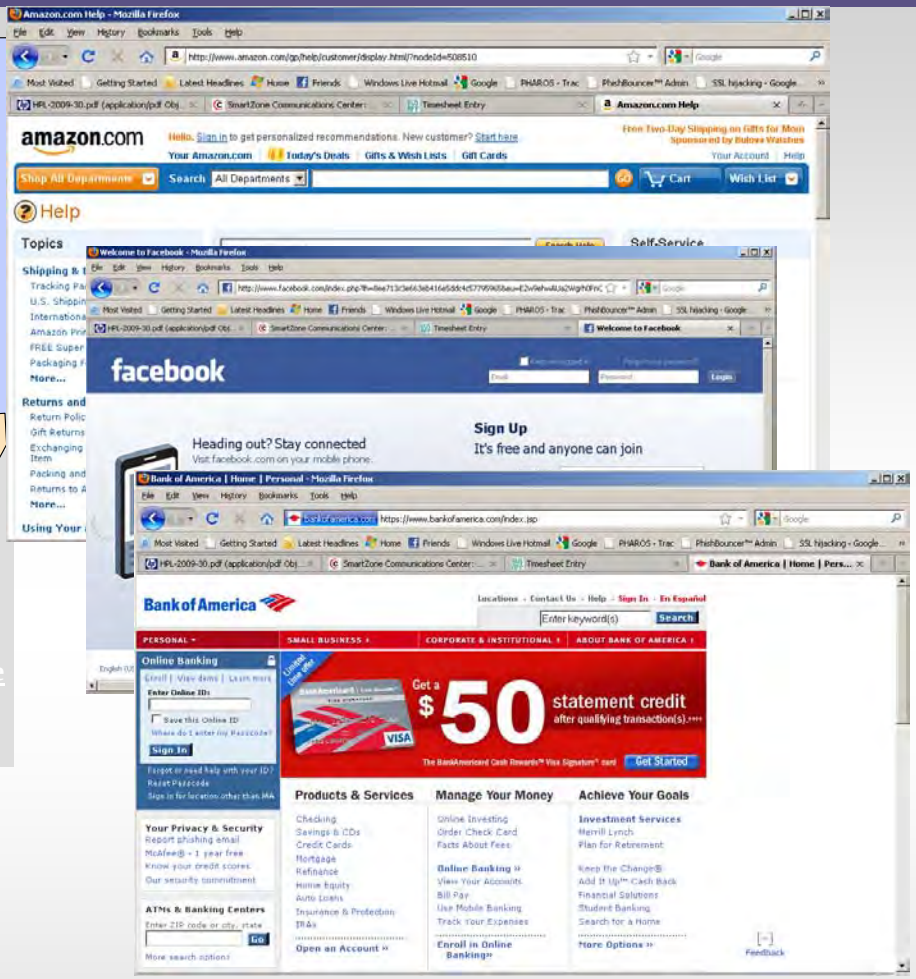
Eventing/Pub-Sub



Streaming/Complex Event Processing

Increasing Role of Information Systems





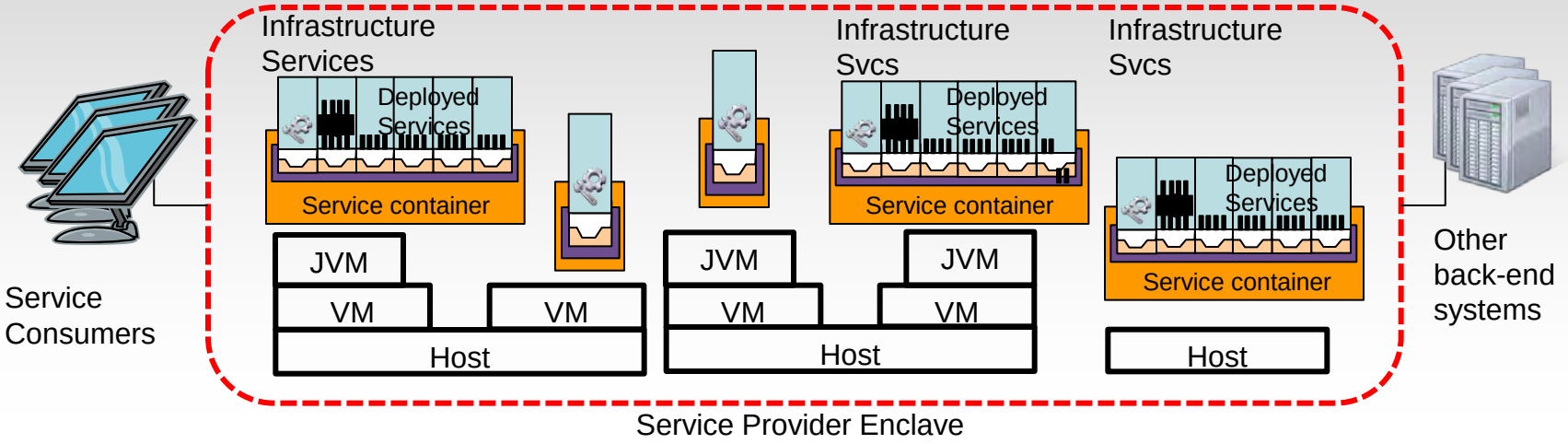
Distributed and Networked Information Systems are increasingly intertwined with military operation as well as civilian life

Unfortunately attacks are on the rise as well

Determined and motivated actors– the Estonia incident, attacks on Israeli systems
Other publicly known instances such as the attacks on DoD systems and defense contractor sites, attacks on Google, recent auction of facebook account information (Kirllos, $45 per 1000 accounts)

SOA Background

Service Oriented Architecture (SOA) facilitates loosely coupled interaction and composition oriented system construction



In a typical SOA offering

- Service Containers – possibly different kinds
- Containers running on virtual machines
- The term “Service Consumers” indicates that they do not offer services to others – deployed services can consume services provided by other deployed or infrastructure services
- Different means of packaging functionality/computation as services such as EJB, Servlets, POJO under WS stack…
- Different ways to access the services e.g., RMI, HTTP/S...
- SOA services may depend on non-SOA back end systems

Many modern information systems, including military systems are migrating to SOA to take advantage of easier enabling of new capabilities and interoperation

Current SOA Services Are Vulnerable

State of the Practice	Limitations
Network &Transport Layer Security (e.g., IPSec, TLS)	Protects network only, ineffective against compromised process or host
Application Layer Security (e.g., DMZs, App Firewalls)	Signature based, static configuration and lack of diversity
Service Infrastructure and Platform Security (e.g., WSS)	Too many standards; a protocol construction kit rather than a solution

Key Security Concerns

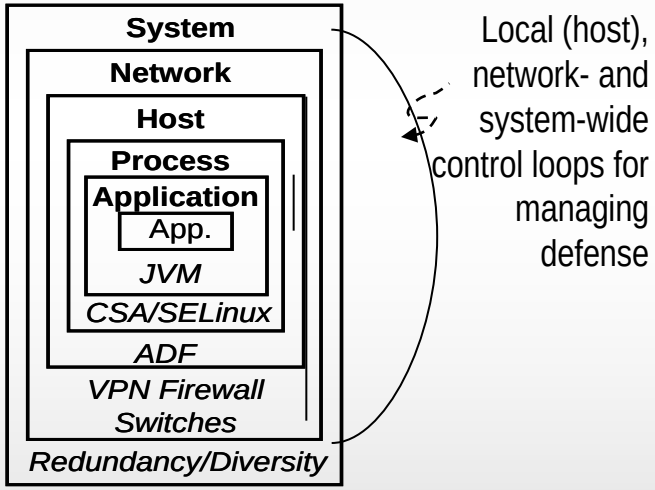
- Usual measures such as Authentication, Access Control, Encryption and Message Signing are not enough for “Fighting Through” attacks from a determined adversary
  - Need to continue to provide service to legitimate consumers
    - Need to contain the attack effects
    - Need to recover (from failures), reconnect (lost connections), regain (lost defenses)
- Tall stacks, complex and large code base – hard to construct an overarching assurance argument using the entire platform as a Trusted Computing Base (TCB)
  - Need to focus on slices, and possibly a root of trust outside the stack

Advanced Protected Services

Need to improve the survivability and protection of next generation service-oriented architectures and systems and applications built using SOA

Current high-water mark in survivable system: OASIS Dem/Val

- Survived 75% of attacks, even when the attacker was given insider access and privilege (red team would actually start the system, after placing attack code)
- A number of survivability design principles that go beyond "defense in depth"
  - Single Point of Failure (SPOF) elimination, redundancy and diversity, containment, hardware or cryptographic root of trust, Crumple zones (many appear in the recent SANS/MITRE Common Weakness Enumeration, CWE)
- Availability is still **the** easiest target, and flaws in COTS components still the major risk (and a fact of life)



Bird's eye view of survivability architecture organizing defense at various system layers

OASIS Dem/Val demonstration was not service-oriented.
APS aims to achieve similar levels of resiliency with SOA

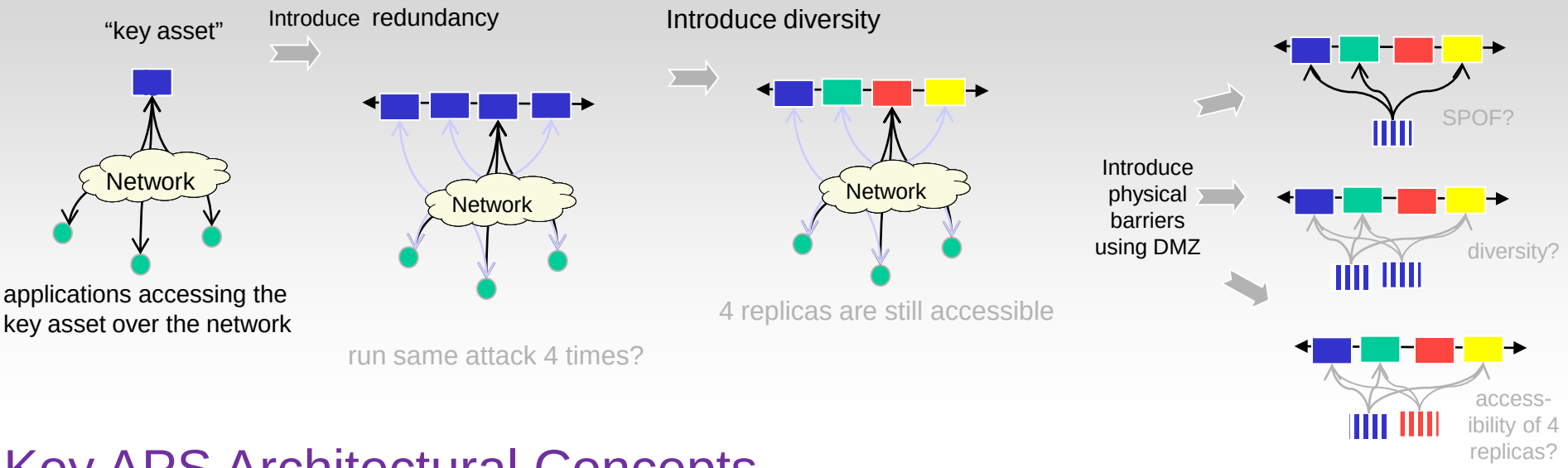
APS Approach

Key Ideas

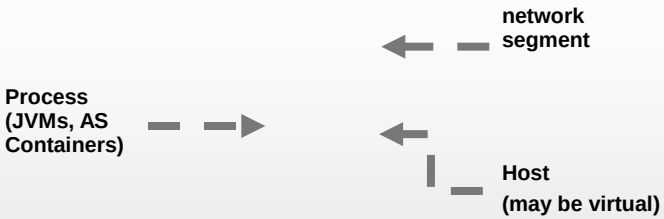
- Extend SOA concepts and design techniques to include elements that facilitate survivable design
  - Specifically focused on “tolerance” or “survival”
  - Strategic concepts like containment and adaptive behavior
- Develop mechanisms, protocols, and supporting services to realize the architecture enhancements
- Develop an environment, and techniques and composition patterns enabling context-specific customization
  - Survivability cost and benefit balanced against the threats and footprint requirements of the deployment environment

Show incremental progress by periodic demonstrations and metrics evaluations, culminating in a red team exercise at the end

Architecting for Survivability

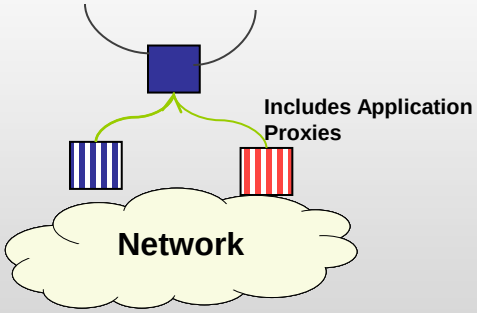


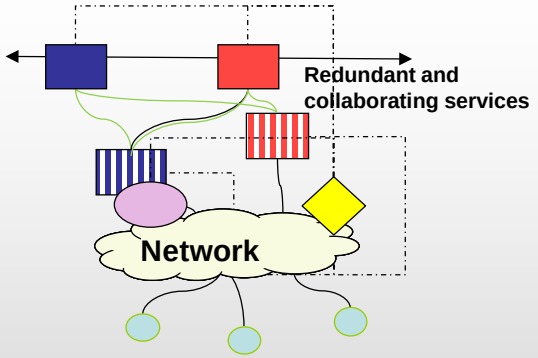
Key APS Architectural Concepts



Containment Regions limit the spread of attacks

Crumple Zones absorb the effects of attacks

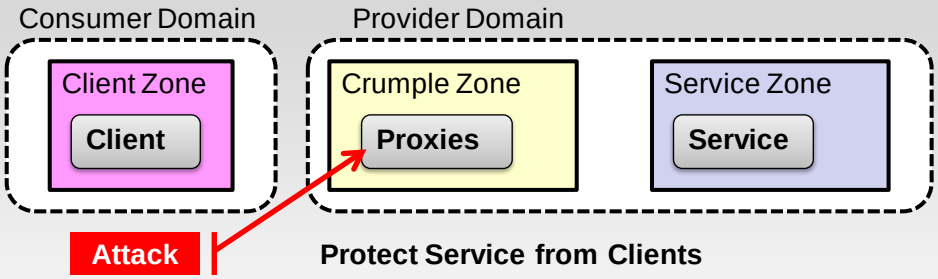




Conglomerates for managed redundancy and collaboration

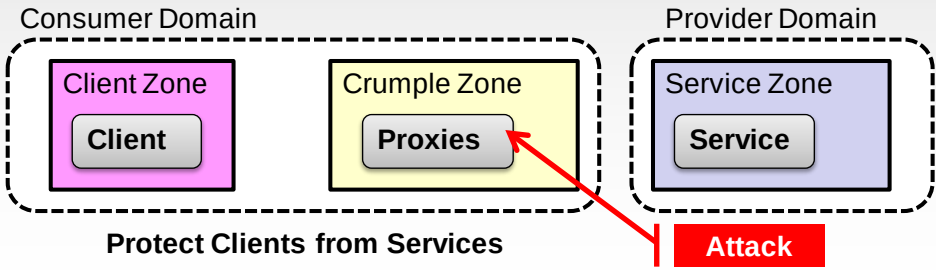
Focusing on Crumple Zone

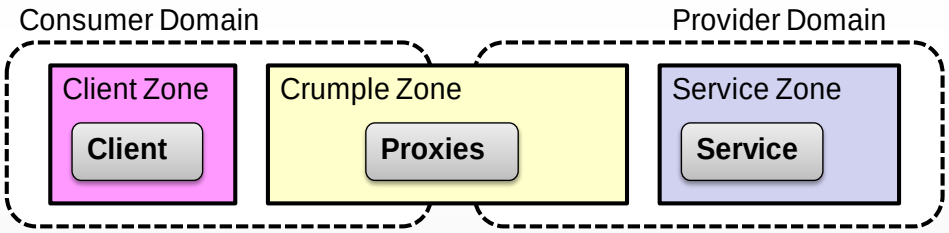
Different Deployment Options



Protecting the Provider Domain

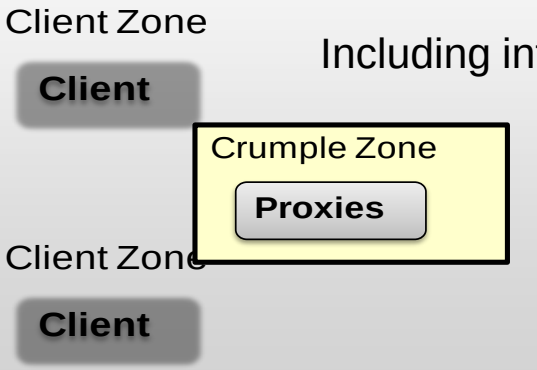
Protecting the Consumer Domain

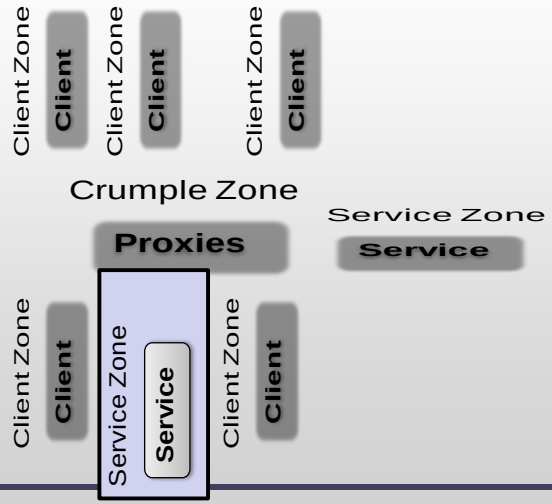




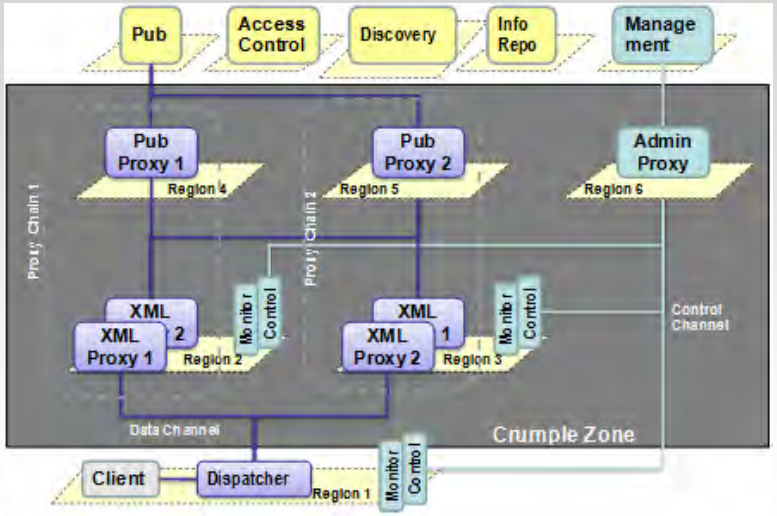
Protecting both Domains

Including inter-client interactions





Crumple Zone: The Concept



Notional depiction of crumple zone

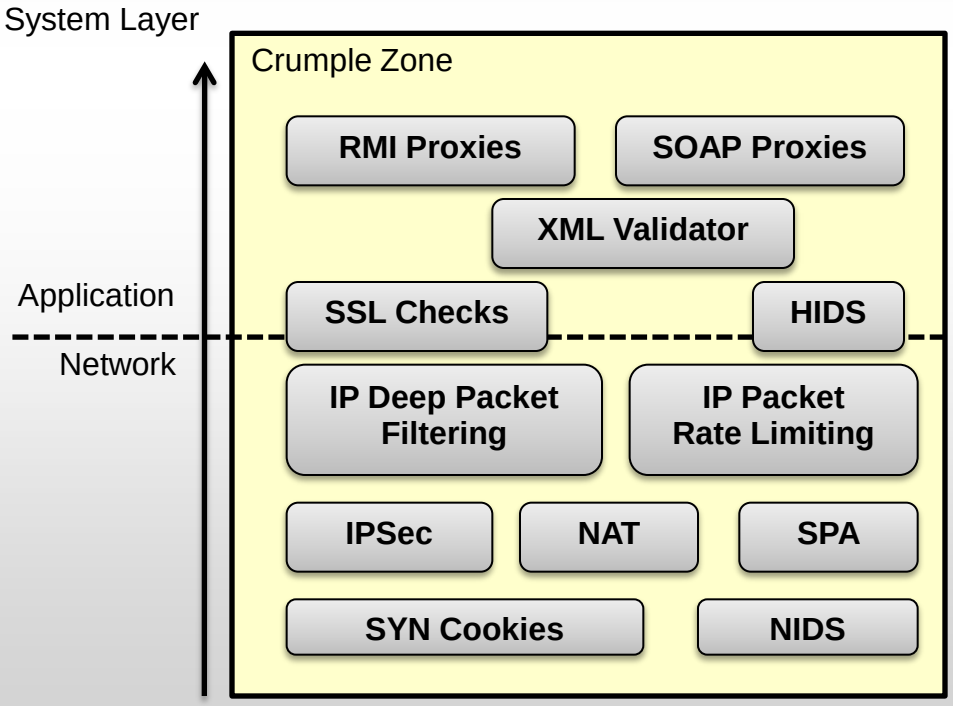
Filtering and checking at various layers
- **Application level (mechanism specific)**
- **Network level**

Maximizing end to end integrity, confidentiality and access control while allowing filtering and checking: sIVPG

Self protection: authentication and access control of control and data within crumple zone

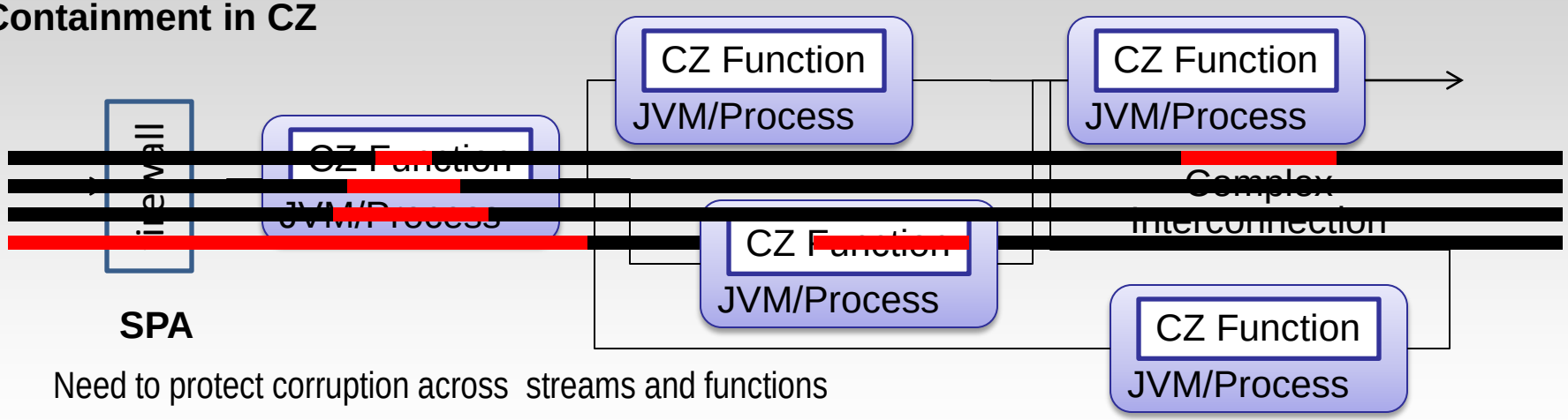
Self management: watchdogs, restarts

Various crumple zone functions with respect to system layers



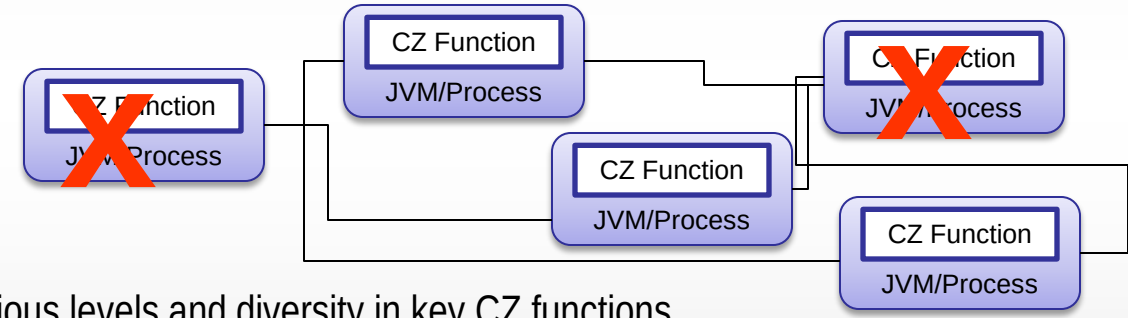
Other APS Concepts at Play in CZ

Containment in CZ



Need to protect corruption across streams and functions

Adaptive response in CZ



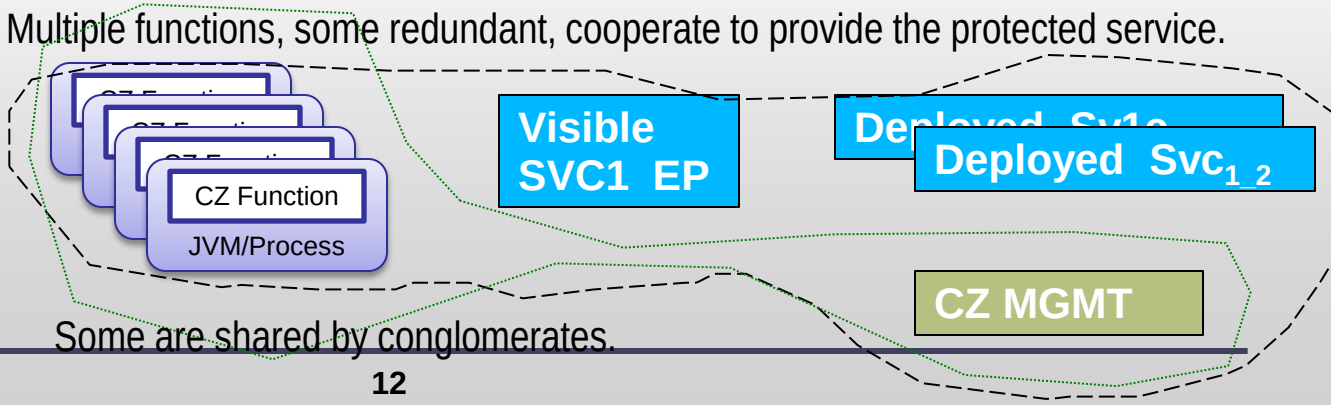
Unavailability of key functions may render the protected services unavailable

Need redundancy, watchdog, restart at various levels and diversity in key CZ functions

CZ as a conglomerate

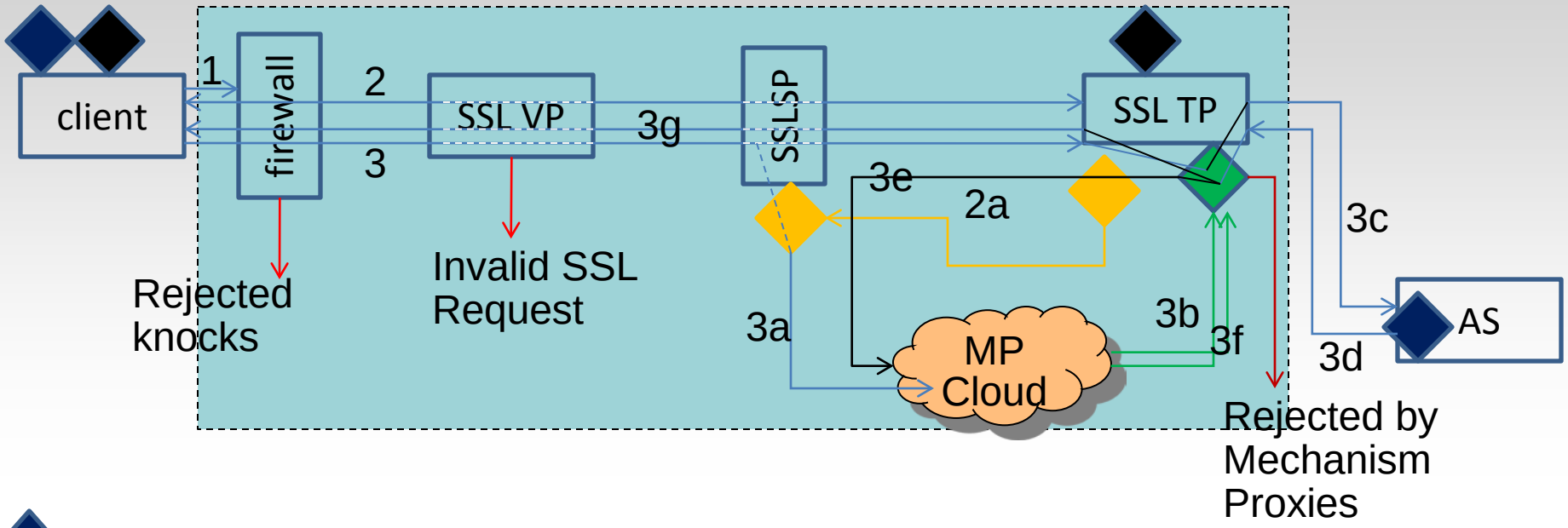
Multiple functions, some redundant, cooperate to provide the protected service.

- Svc1 Consumer$_1$
- Svc1 Consumer$_2$
- Svc1 Consumer$_n$



Some are shared by conglomerates.

One Realization (SSL, EJB)



- Service requester and provider
- SSL end point function
- Key sharing function
- Escrow and decision function

E2E integrity envelope is not all the way, but minimal key sharing and control

- Split done within the message layer, i.e., RMI

Easier handling of escrow and decision for both request and response

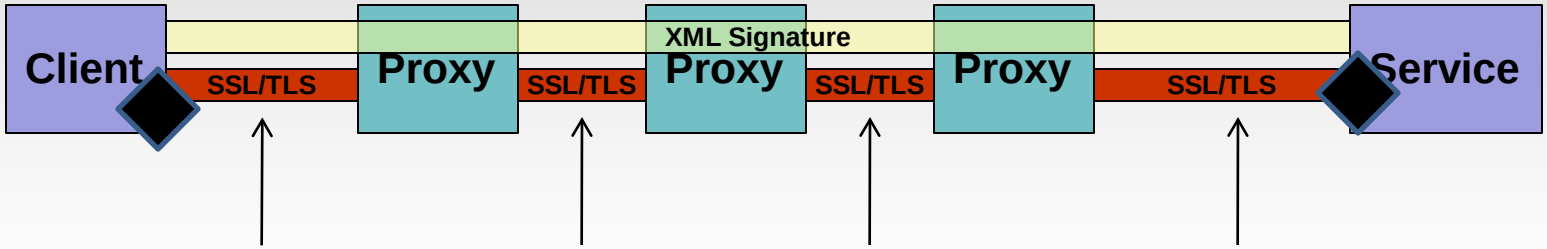
Multiple possibilities for situating the escrow

Different configurations have different assurance guarantees – all using the same basic building block functions…

Mechanism in Support of CZ: slVPG

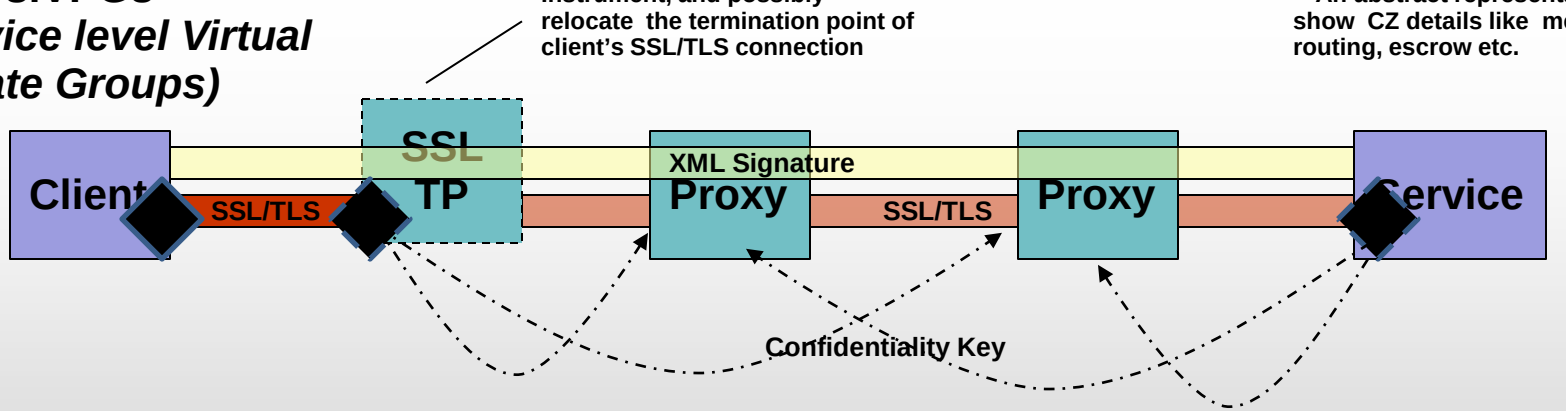
JSSWG Control Case E: SSL/TLS + XML Signature

Inspection and Filtering in the Crumple Zone Without slVPGs



- **Multiple connections imply new Proxy certificates and less flexible Proxy configuration**

With slVPGs*
(*service level Virtual Private Groups*)

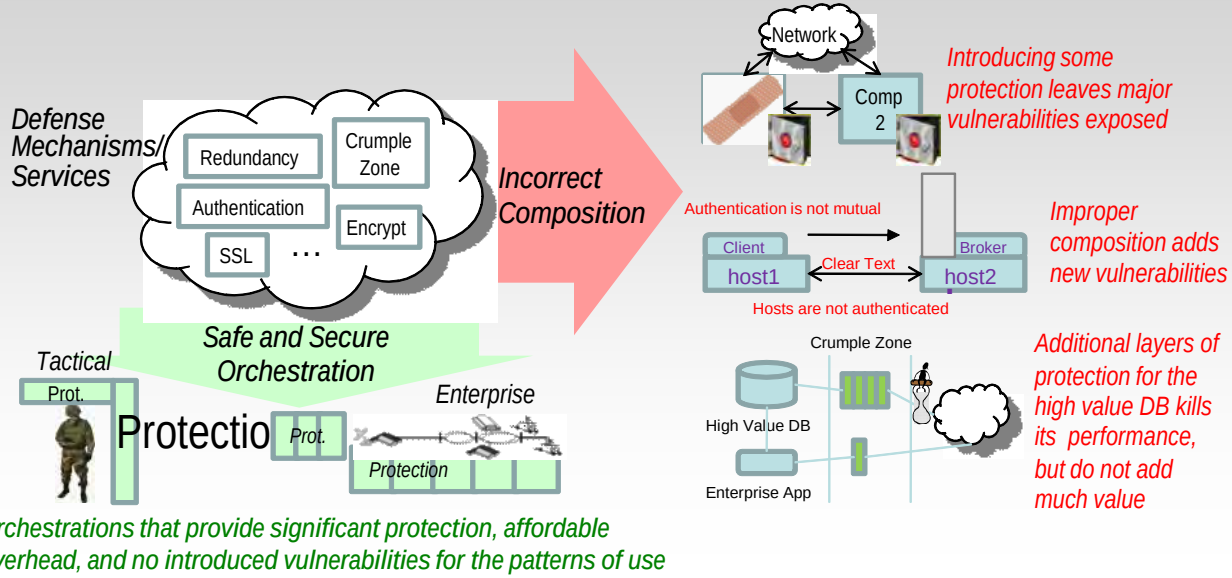


- **In effect, the proxies are able to see the messages in clear text**
- **Proxies do not require their own certificates**

Support for Safe Composition

Without help for composition:

- Expecting some defense to be a *silver bullet*, but leaving major vulnerabilities unprotected
- Inadvertently introducing vulnerabilities during composition
- Including defenses that provide little benefit at too great a cost



- Define a set of operational use cases, identify the vulnerabilities and footprint, performance, and overhead constraints of the use case
- Develop *benefit/cost* ratios for each protection technique and use case
- Rank the techniques according to their benefit/cost ratio, taking into account the additional vulnerabilities and dependencies on other techniques
- Define and enforce policies/contracts on the composition, using automated checkers
- Define automated configuration generators from high level specifications of the composition patterns and characteristics of the use cases

Achievements So Far

- Assessment of ONR RI VMs
- Reviewed a set of security community guidance documents
- APS requirements
- APS platform (baseline)
- Survivability enhancements
  - Design, Implementation, Testing and Evaluation

Ongoing Tasks

Conclusion

- Goal: engineering of survivable service-based systems
  - Robust and repeatable methodology
  - Catalog of building blocks with cost-benefit annotation
  - Best practices
- Defending against a sophisticated adversary is difficult, critical information systems need to be more resilient under a wider range of hostile environments and contested situations
- No protection is absolute, but with the right combination of sound engineering and innovative techniques, we can raise the bar pretty high
  - As demonstrated in OASIS Dem/Val
- Work in progress, with good initial results
  - We just successfully demonstrated a version of the crumple zone